DATA PRIVACY AND SECURITY

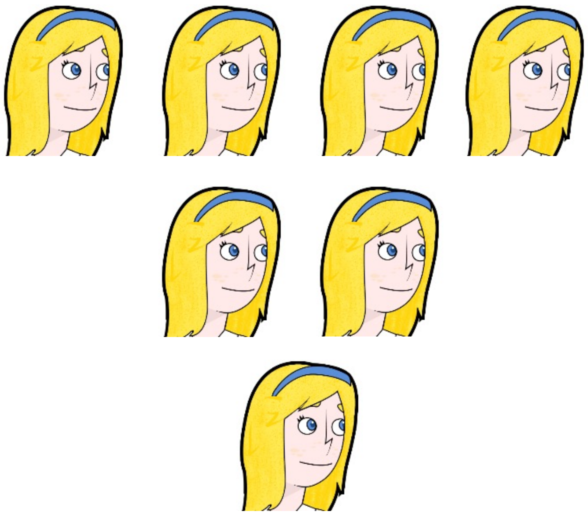
Prof. Daniele Venturi

Master's Degree in Data Science
Sapienza University of Rome

CIS SAPIENZA
RESEARCH CENTER FOR CYBER INTELLIGENCE AND INFORMATION SECURITY

CHAPTER 4: Big Data & Cloud Cryptography

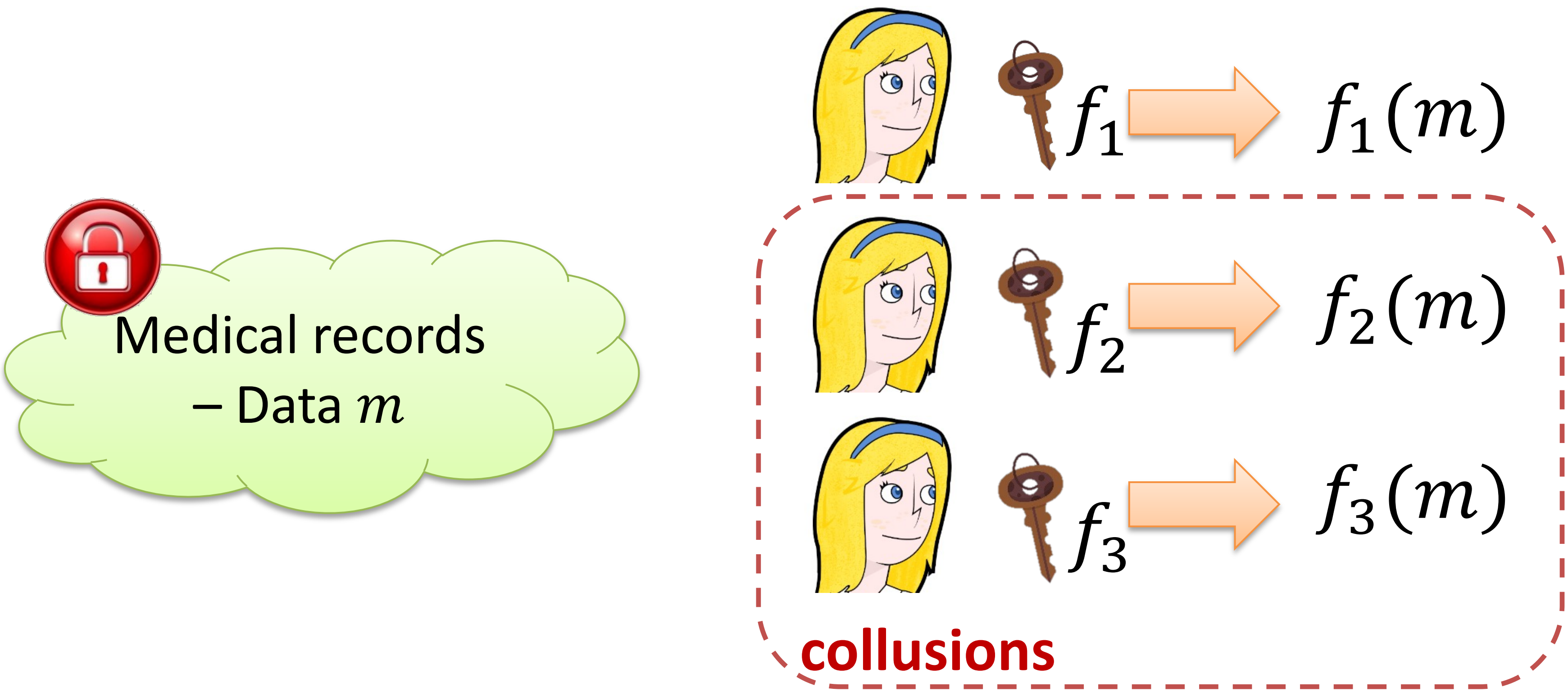
Financial, medical, customers, employees



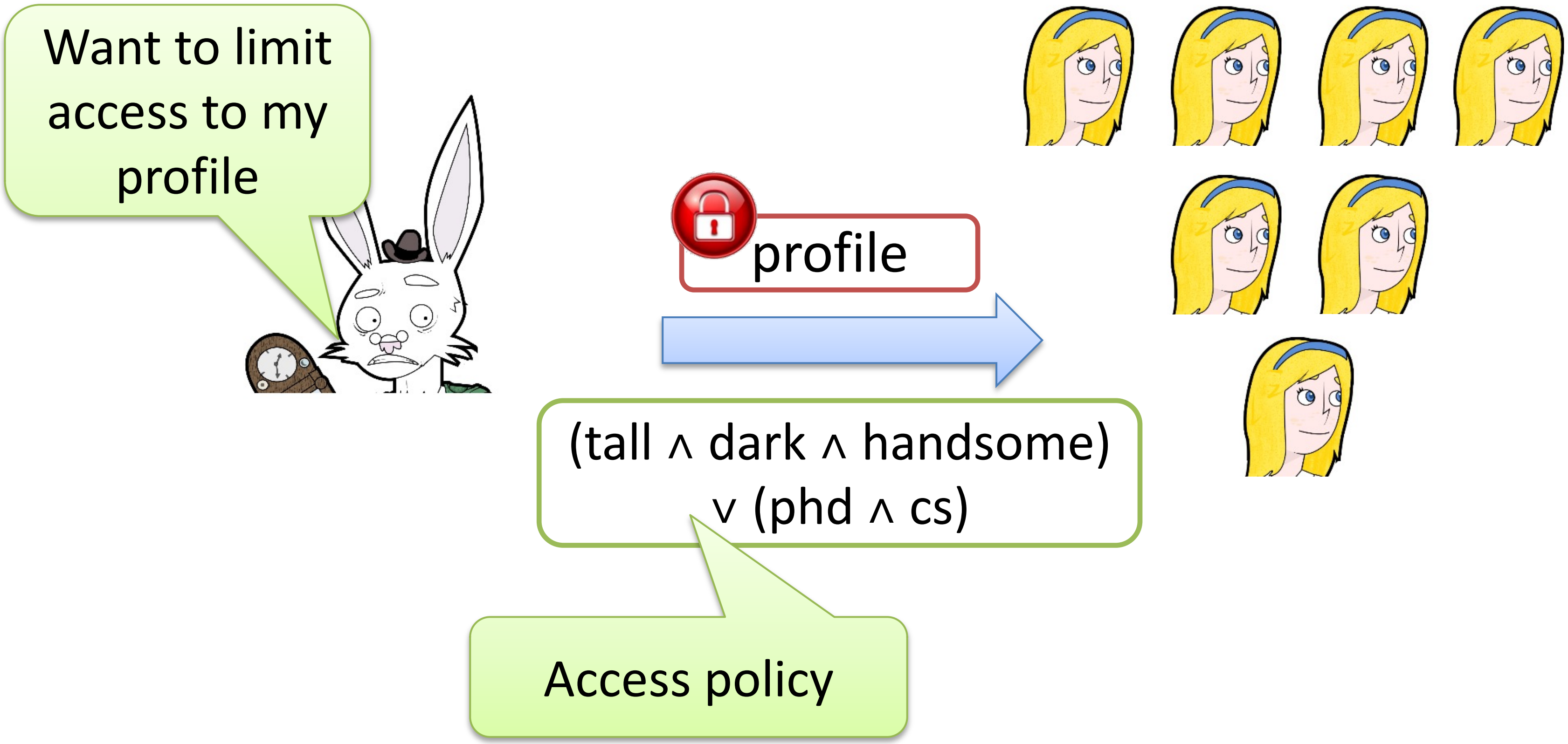
BIG DATA

- **Utility + privacy**
  - Restrict access
  - Restrict computation

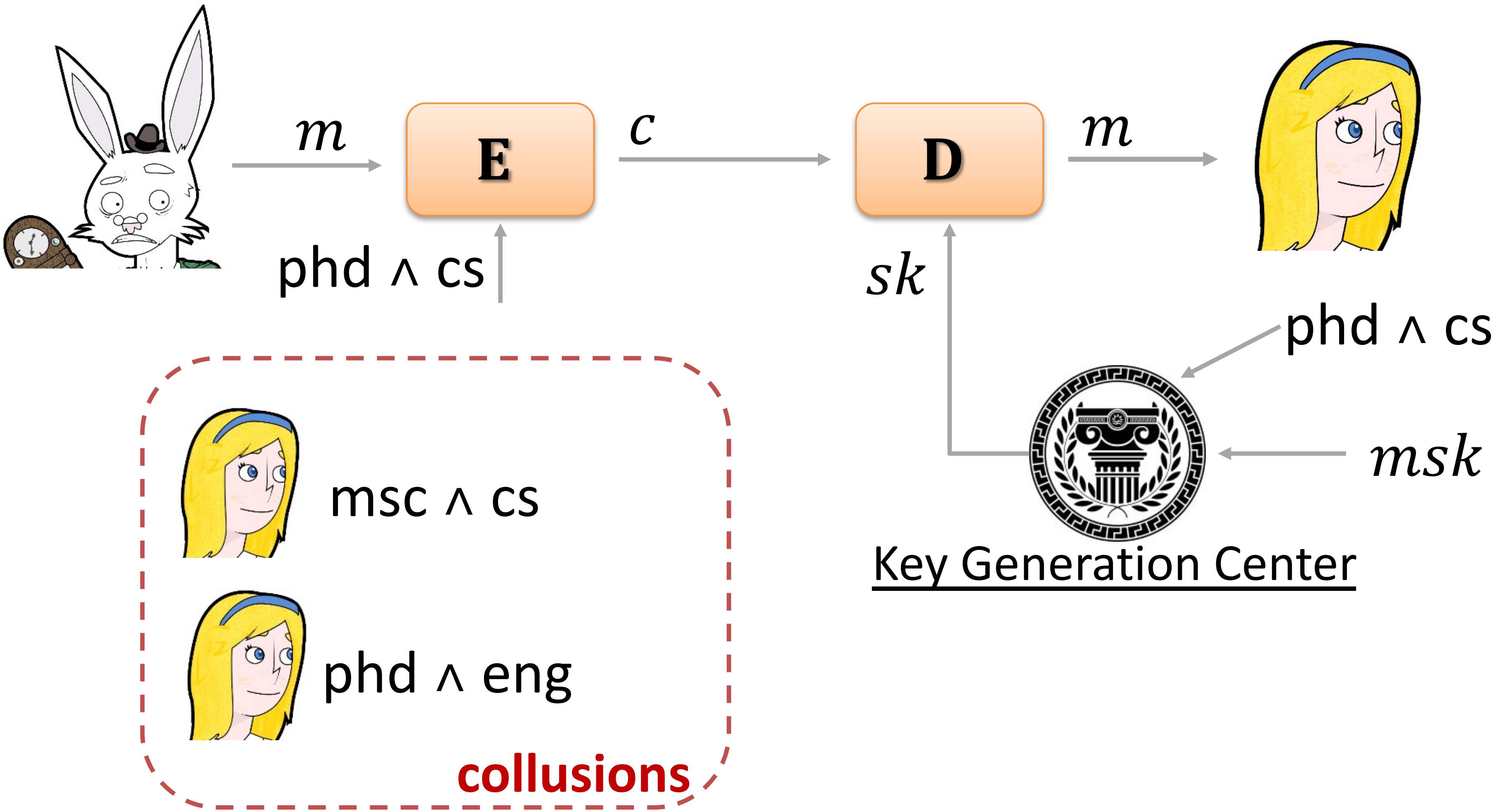
Functional Encryption (FE)



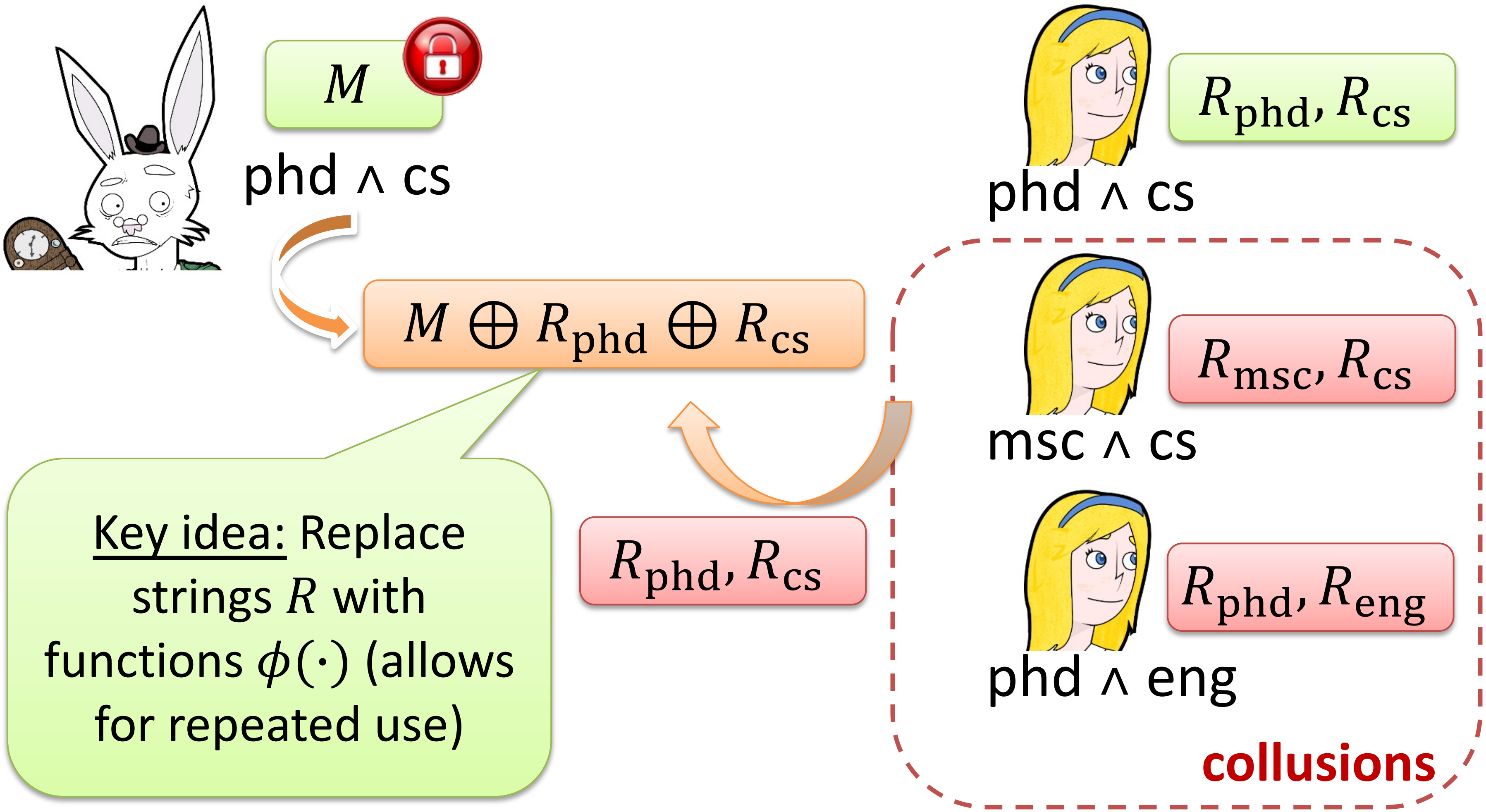
Dating and Big Data



Attribute-Based Encryption (ABE)



Mix-and-Match Attacks

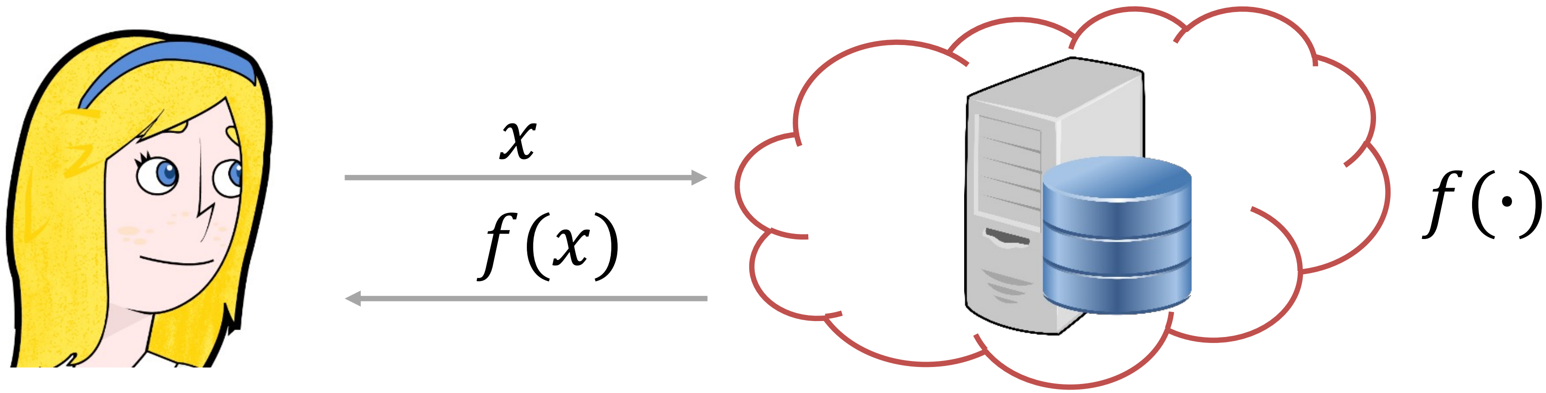


Results on FE and ABE

- Constructions of FE for **arbitrary functions** currently requires **strong assumptions**
  - Multi-linear maps
  - Indistinguishability obfuscation
- The situation is much better for ABE
  - Constructions for **arbitrary policies** from LWE
  - Constructions for **arbitrary policies** using pairings

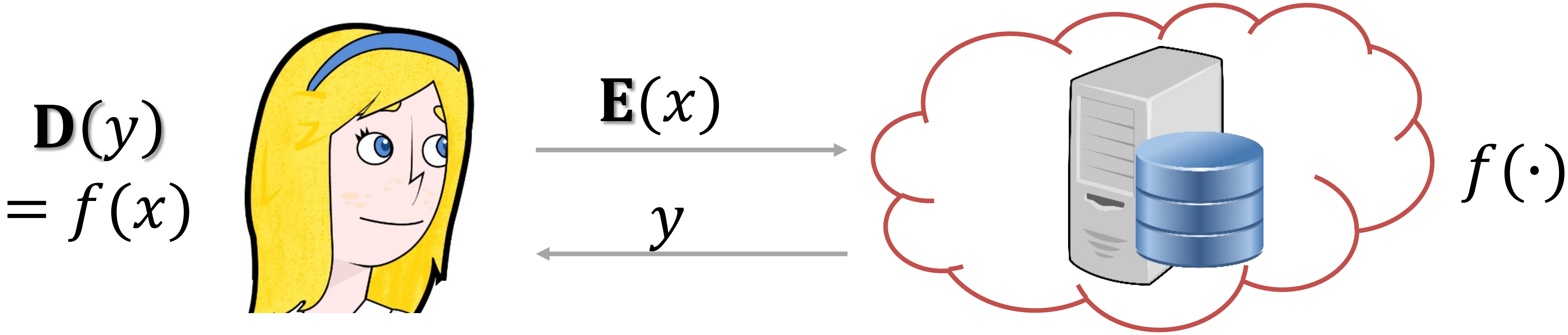
> *"Cryptographers seldom sleep well"* – Silvio Micali

Outsourcing of Computation



- Email, web search, navigation, social networking, …
- What about **private** x?

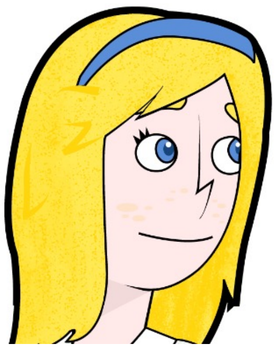
Outsourcing of Computation - Privately



WISH: Homomorphic evaluation function:

$$\mathbf{C}: f, \mathbf{E}(x) \rightarrow \mathbf{E}(f(x))$$

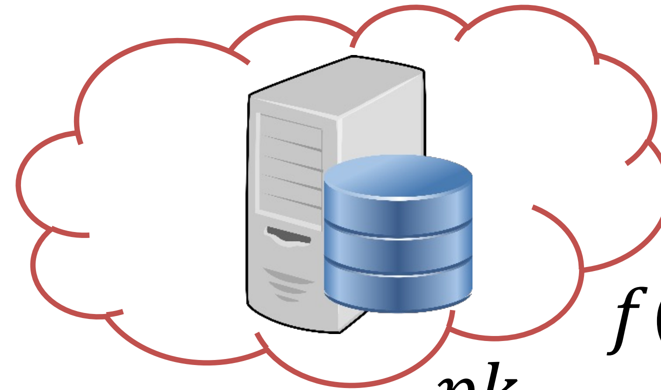
Fully Homomorphic Encryption



pk, sk

$c = \mathbf{E}(pk, x)$ →

$y = \mathbf{C}(pk, f, c)$ ←



$f(\cdot)$

pk

Correctness:

$$\mathbf{D}(sk, y) = f(x)$$

Privacy:

$$\mathbf{E}(pk, x) \approx \mathbf{E}(pk, 0)$$

FHE = Correctness $\forall$ efficient f = Correctness for universal set

Levelled FHE: Bounded depth f

- NAND
- $(+,\times)$ over a ring

Trivial FHE?

- Let $(\mathbf{E}, \mathbf{D})$ be any PKE scheme
- Define FHE $(\mathbf{E}', \mathbf{D}', \mathbf{C}')$:
  - $\mathbf{E}'$ identical to $\mathbf{E}$
  - $\mathbf{C}'(pk, f, c) = (f, c)$
  - $\mathbf{D}'(sk, c) = f(\mathbf{D}(c))$

Compact FHE: ∃ **global bound** on ciphertext length and decryption time

A Paradox (And its Resolution)

$$c_1 = \mathbf{E}(pk, x_1)$$
$$c_2 = \mathbf{E}(pk, x_2)$$
$$c_3 = \mathbf{E}(pk, x_3)$$

$$f(x_1, x_2, x_3) = \begin{cases} x_2 \text{ if } x_1 = 0 \\ x_3 \text{ if } x_1 = 1 \end{cases}$$

$$\mathbf{E}(pk, x_2)$$

$$\mathbf{C}(pk, f, c_1, c_2, c_3)$$

AH! So
$x_1 = 0$

- But remember that encryption is **randomized**!
- Output of evaluation algorithm will look as a **fresh** and **random** ciphertext

Eigenvectors Method (Basic Idea)

- Let C_1 and C_2 be matrixes for **eigenvector** $\vec{s}$, and **eigenvalues** x_1, x_2 (i.e., $\vec{s} \times C_i = x_i \cdot \vec{s}$)
  - $C_1 + C_2$ has eigenvalue $x_1 + x_2$ w.r.t. $\vec{s}$
  - $C_1 \times C_2$ has eigenvalue $x_1 \cdot x_2$ w.r.t. $\vec{s}$
- Idea (GSW): Let C be the ciphertext, $\vec{s}$ be the secret key and x be the plaintext (say over $\mathbb{Z}_q$)
  - Useful to think of $\mathbb{Z}_q = [-q/2, q/2)$
  - Homomorphism for **addition/multiplication**
  - But **insecure**: Easy to compute eigenvalues

Approximate Eigenvectors Method

- Approximate variant: $\vec{s} \times C = x \cdot \vec{s} + \vec{e} \approx x \cdot \vec{s}$
  - "Decryptable" as long as $\|\vec{e}\|_\infty \ll q$

$$\vec{s} \times C_1 = x_1 \cdot \vec{s} + \vec{e}_1 \qquad \|\vec{e}_1\|_\infty \ll q$$

$$\vec{s} \times C_2 = x_2 \cdot \vec{s} + \vec{e}_2 \qquad \|\vec{e}_2\|_\infty \ll q$$

- **Goal:** Define **homomorphic** operations

$C_{\text{add}} = C_1 + C_2$:

$$\vec{s} \times (C_1 + C_2) = \vec{s} \times C_1 + \vec{s} \times C_2$$
$$= x_1 \cdot \vec{s} + \vec{e}_1 + x_2 \cdot \vec{s} + \vec{e}_2$$
$$= (x_1 + x_2) \cdot \vec{s} + (\vec{e}_1 + \vec{e}_2)$$

Noise grows a little!

Approximate Eigenvectors Method

- Approximate variant: $\vec{s} \times C = x \cdot \vec{s} + \vec{e} \approx x \cdot \vec{s}$
  - "Decryptable" as long as $\|\vec{e}\|_\infty \ll q$

$$\vec{s} \times C_1 = x_1 \cdot \vec{s} + \vec{e}_1 \qquad \|\vec{e}_1\|_\infty \ll q$$

$$\vec{s} \times C_2 = x_2 \cdot \vec{s} + \vec{e}_2 \qquad \|\vec{e}_2\|_\infty \ll q$$

- **Goal:** Define **homomorphic** operations

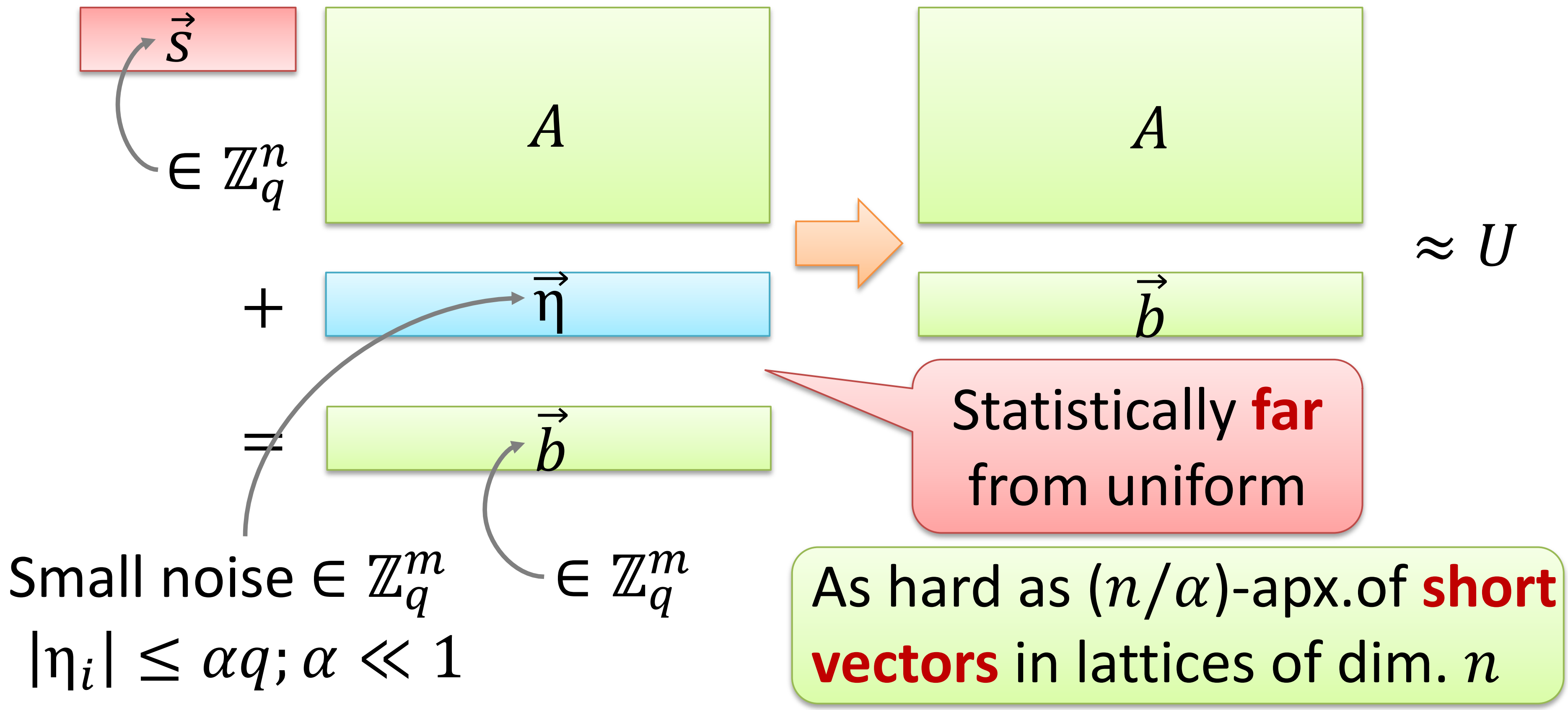
$C_{\text{mult}} = C_1 \times C_2$:

$$\begin{aligned} \vec{s} \times (C_1 \times C_2) &= (x_1 \cdot \vec{s} + \vec{e}_1) \times C_2 \\ &= x_1 \cdot (x_2 \cdot \vec{s} + \vec{e}_2) + \vec{e}_1 \times C_2 \\ &= x_1 \cdot x_2 \cdot \vec{s} + (x_1 \cdot \vec{e}_2 + \vec{e}_1 \times C_2) \end{aligned}$$

Noise grows! Needs to be small!

Learning with Errors (LWE)

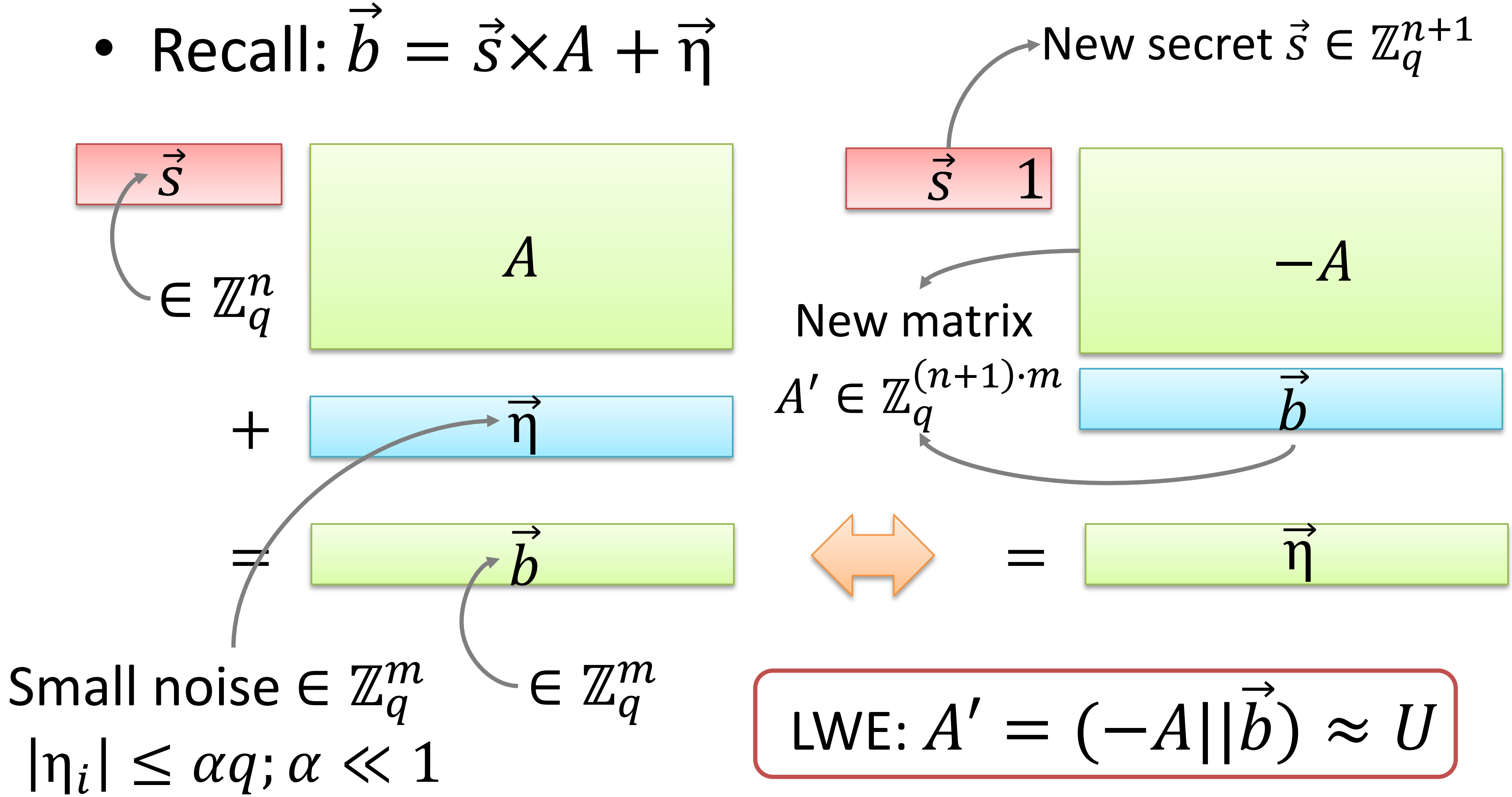
- Random **noisy** linear equations ≈ uniform



CIS SAPIENZA
Research Center for Cyber Intelligence and Information Security

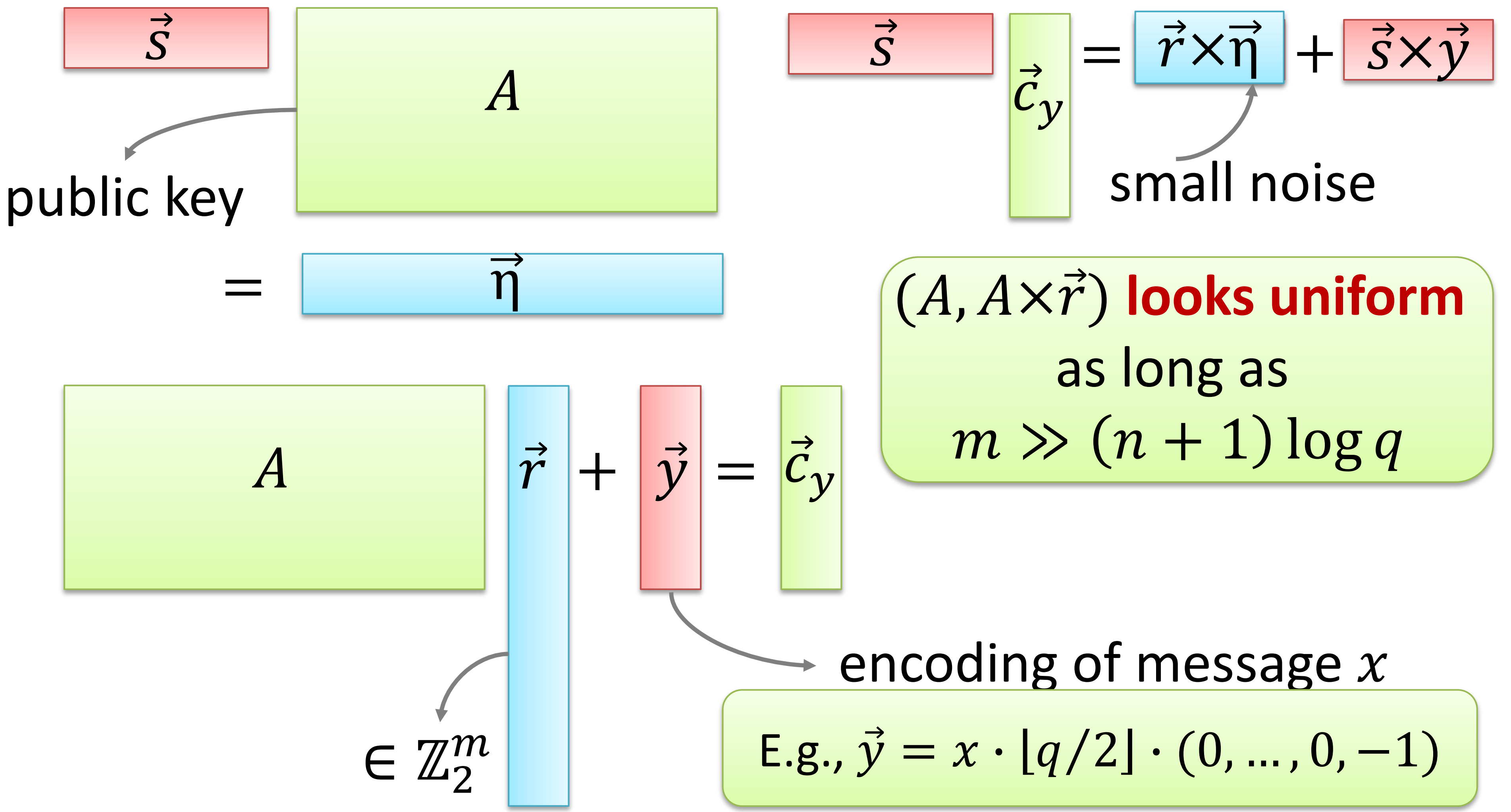
LWE – Rearranging Notation

- Recall: $\vec{b} = \vec{s} \times A + \vec{\eta}$



LWE: $A' = (-A||\vec{b}) \approx U$

PKE from LWE


$\vec{s}$
A
public key
$= \vec{\eta}$
$\vec{s}\ \vec{c}_y = \vec{r}\times\vec{\eta} + \vec{s}\times\vec{y}$
small noise
$(A, A\times\vec{r})$ **looks uniform**
as long as
$m \gg (n+1)\log q$
$A\ \vec{r} + \vec{y} = \vec{c}_y$
$\in \mathbb{Z}_2^m$
encoding of message x
E.g., $\vec{y} = x\cdot\lfloor q/2\rfloor\cdot(0,\dots,0,-1)$

PKE from LWE – Matrix Version

$$A \cdot R + Y = C_Y$$

$R \in \mathbb{Z}_2^{m \cdot N}$

$Y \in \mathbb{Z}_q^{(n+1) \cdot N}$ encoding of message

$$\vec{s} \cdot C_Y = \vec{\eta} \times \mathrm{R} + \vec{s} \times Y$$

$\vec{\eta} \times \mathrm{R}$: small noise

Shrinking Gadgets

- Write entries in C using **binary decomposition**

$$C = \begin{bmatrix} 3 & 5 \\ 1 & 4 \end{bmatrix} \pmod 8 \xrightarrow{\text{yields}} \text{bits}(C) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \pmod 8$$

small entries!

- Reverse operation:

$$\Rightarrow \vec{s} \times C = \vec{s} \times G \times G^{-1}(C)$$

$k \cdot N = k\lceil \log q \rceil$

k

$$\begin{bmatrix} 2^{N-1} & \dots & 2 & 1 & 0 & \dots & 0 & 0 \\ 0 & \dots & 0 & 0 & 2^{N-1} & \dots & 2 & 1 \end{bmatrix} \times \text{bits}(C) = C$$

G $\quad$ $G^{-1}(C)$

The GSW Scheme

$\vec{s}$

A

public key $\in \mathbb{Z}_q^{n \cdot m} =$

$\vec{\eta}$

$\in \mathbb{Z}_2^{m \cdot N} = \mathbb{Z}_2^{m\lceil \log q \rceil}$

$$\mathbf{E}(A, x; R) = A \times R + x \cdot G = C_{x \cdot G}$$

$$\|\vec{e}\|_\infty = \|\vec{\eta} \times R\|_\infty \leq n \cdot m$$

Invariant: $\vec{s} \times C = \vec{e} + x \cdot \vec{s} \times G$

$$\mathbf{D}(\vec{s}, C) = \vec{s} \times C \times G^{-1}(-\lfloor q/2 \rfloor \cdot \vec{u}_{n+1})$$
$$= \vec{e} \times G^{-1}(\cdots) + x \cdot \vec{s} \times G \times G^{-1}(-\lfloor q/2 \rfloor \cdot \vec{u}_{n+1})$$
$$= \vec{e} \times G^{-1}(\cdots) + \lfloor q/2 \rfloor \cdot x$$

Output: $0 \Leftrightarrow |\mathbf{D}(\vec{s}, C)| < q/4$

The GSW Scheme – Homomorphism

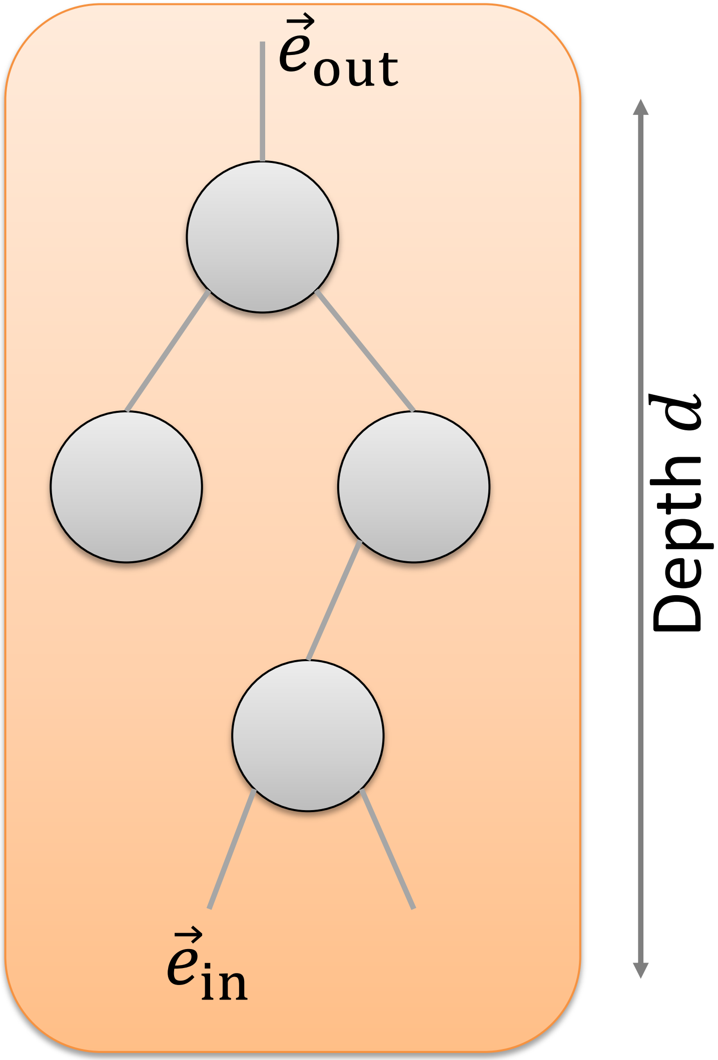
Invariant: $\vec{s} \times C = \vec{e} + x \cdot \vec{s} \times G$

$$C_{\text{mult}} = C_1 \times G^{-1}(C_2)$$

$$\begin{aligned}
&\vec{s} \times C_1 \times G^{-1}(C_2) = (\vec{e}_1 + x_1 \cdot \vec{s} \times G) \cdot G^{-1}(C_2) \\
&= \vec{e}_1 \times G^{-1}(C_2) + x_1 \cdot \vec{s} \times G \times G^{-1}(C_2) \\
&= \vec{e}_1 \times G^{-1}(C_2) + x_1 \cdot \vec{s} \times C_2 \\
&= \vec{e}_1 \times G^{-1}(C_2) + x_1 \cdot (\vec{e}_2 + x_2 \cdot \vec{s} \times G) \\
&= (\vec{e}_1 \times G^{-1}(C_2) + x_1 \cdot \vec{e}_2) + x_1 x_2 \cdot \vec{s} \times G \\
&= \vec{e}_{\text{mult}} + x_1 x_2 \cdot \vec{s} \times G
\end{aligned}$$

$$||\vec{e}_{\text{mult}}||_\infty \leq N \cdot ||\vec{e}_1||_\infty + ||\vec{e}_2||_\infty \leq (N+1) \cdot \max\{||\vec{e}_1||, ||\vec{e}_2||\}$$

Homomorphic Circuit Evaluation



$$\|\vec{e}_{\text{out}}\|_\infty \leq (N+1)^{d+1} m \cdot \alpha q$$

Decryptability:

$$n \cdot m \cdot (N+1)^{d+1} < q/4$$

Security: $m \geq 1 + 2n(2 + \log q)$

and $q \leq 2^{n^\epsilon}$ $(\epsilon < 1)$

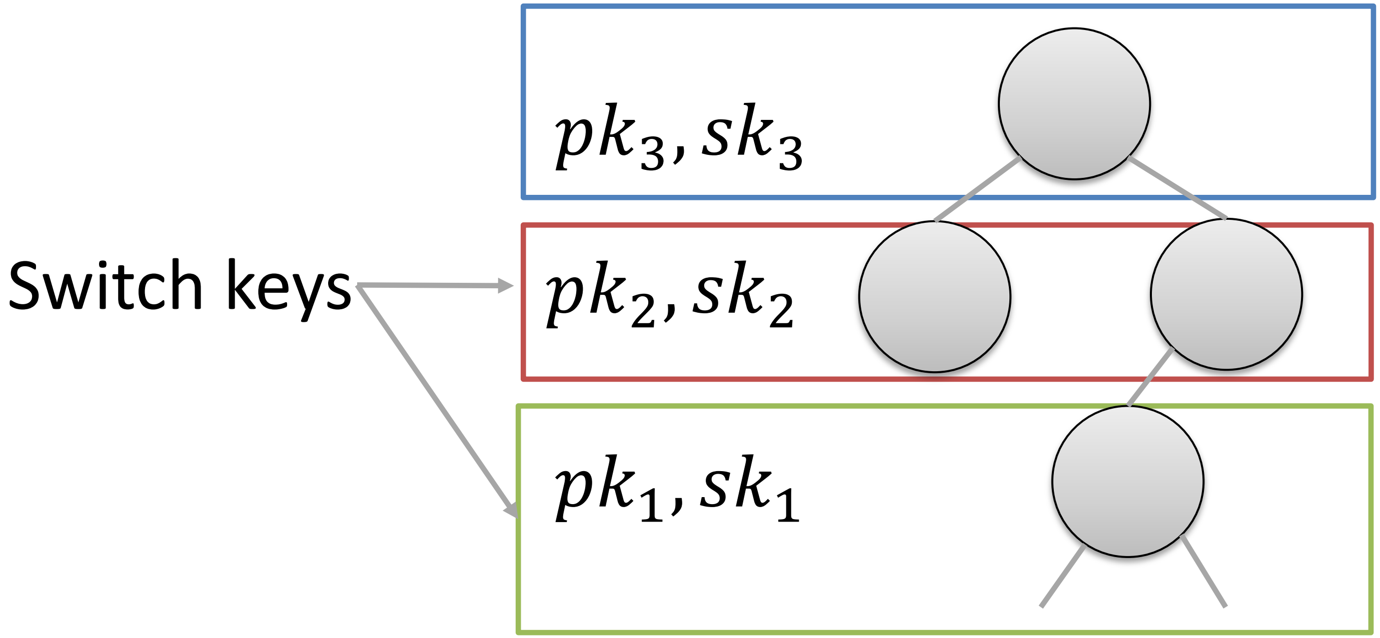
$\Rightarrow n^\epsilon > 2d \cdot \log n$

$$\|\vec{e}_{\text{i+1}}\|_\infty \leq (N+1)\|\vec{e}_{\text{i}}\|_\infty$$

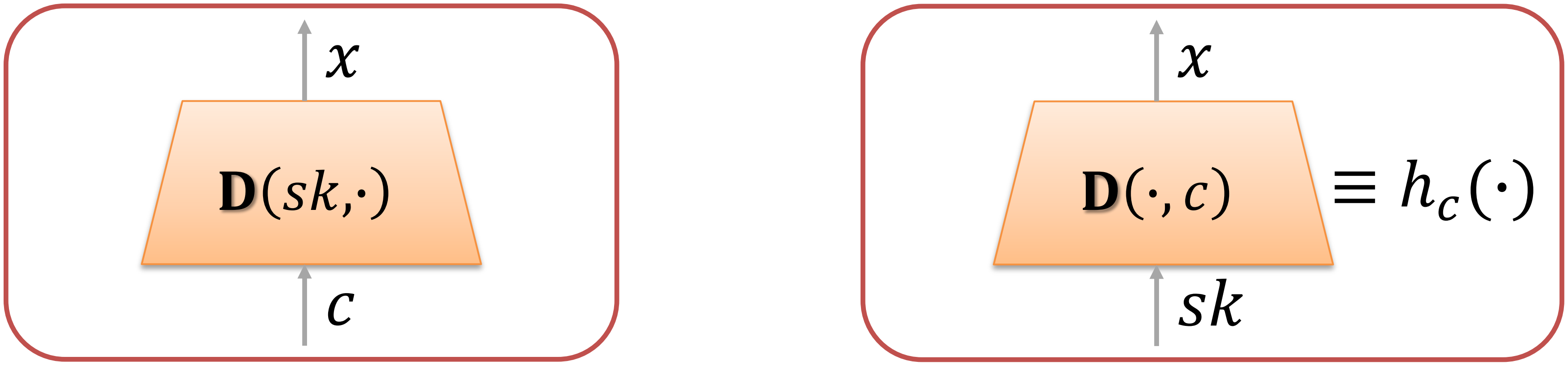
$$\|\vec{e}_{\text{in}}\|_\infty \leq m \cdot n = m \cdot \alpha q$$

Bootstrapping

- Given scheme with **bounded homomorphism** up to d_{hom}, can we **extend** its homomorphic capability?
- Idea: Do a few operations, then **switch key**



How to Switch Keys



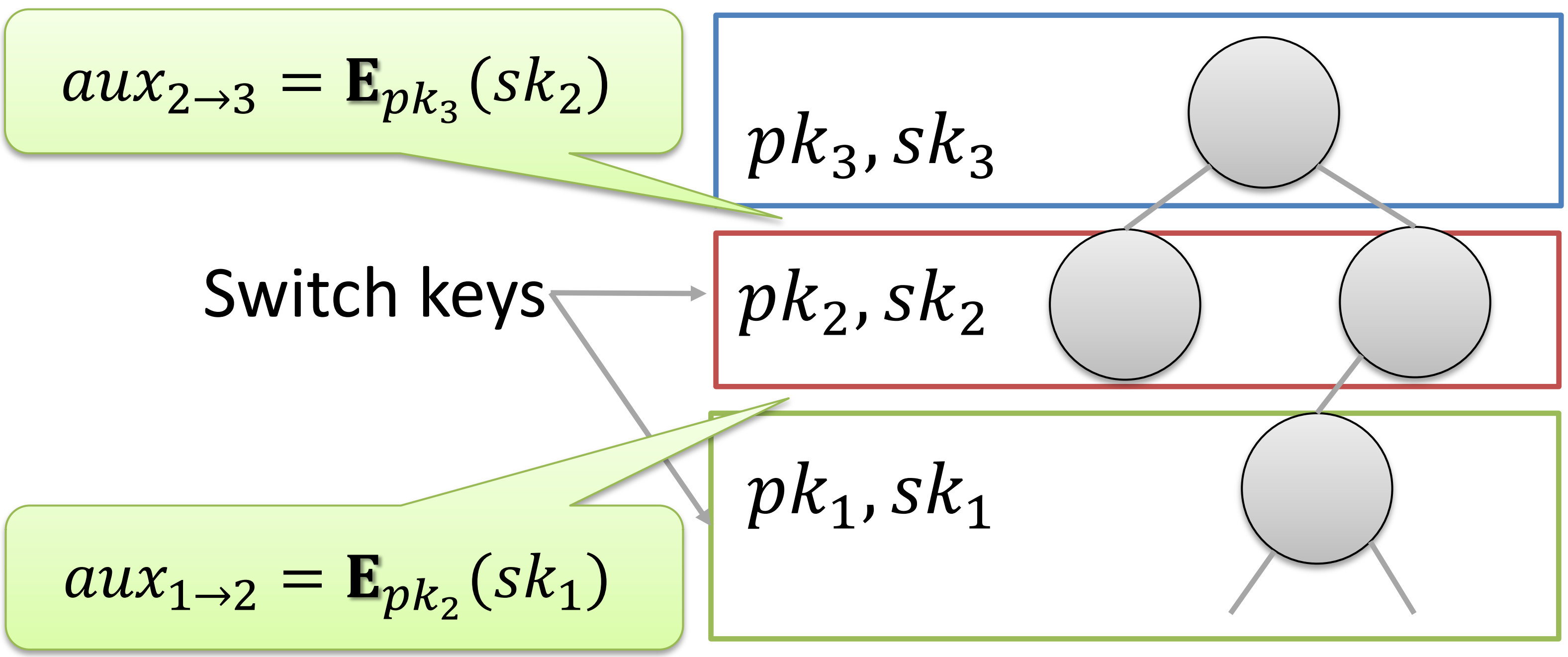
Decryption circuit

Dual view

$$\begin{aligned}\mathbf{C}_{pk'}(h_c, aux) &= \mathbf{C}_{pk'}\left(h_c, \mathbf{E}_{pk'}(sk)\right) \\ &= \mathbf{E}_{pk'}(h_c(sk)) \\ &= \mathbf{E}_{pk'}(x)\end{aligned}$$

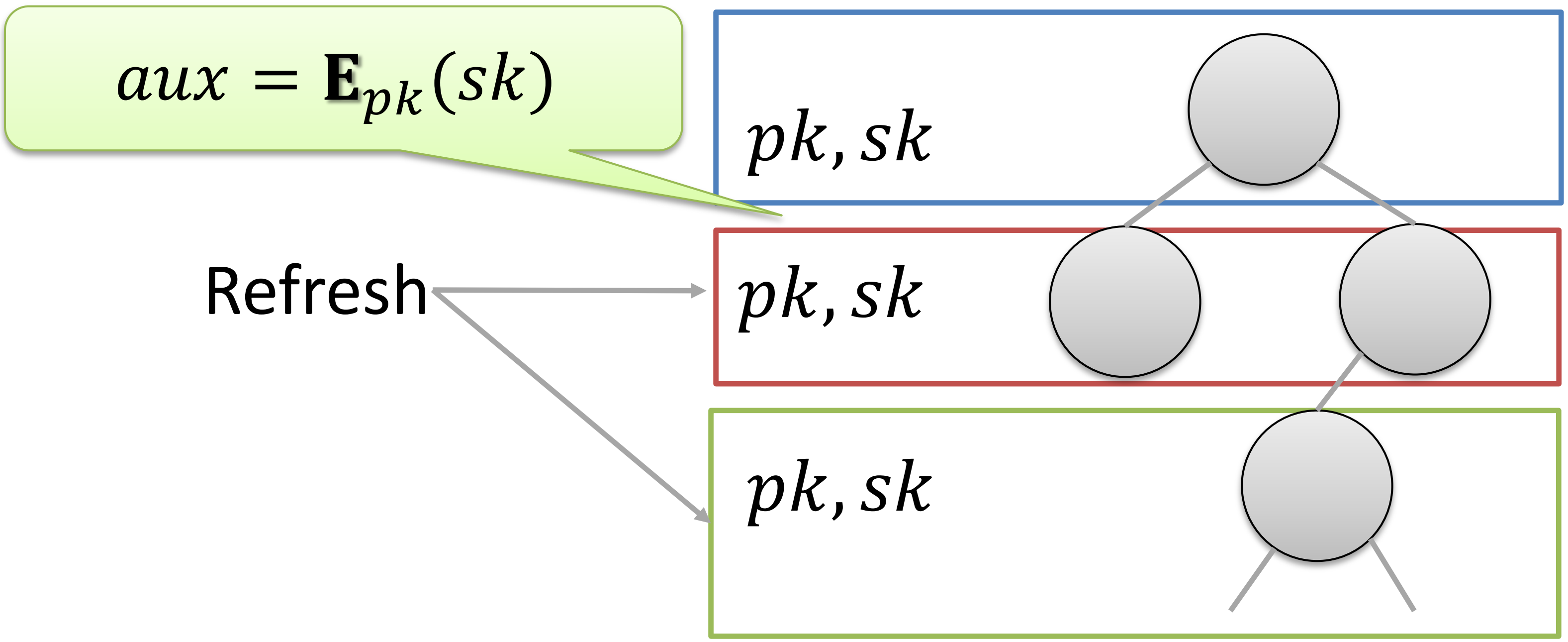
Bootstrapping Theorem

- Homomorphic capacity of output: $d_{\text{hom}} - d_{h_c} = d_{\text{hom}} - d_{\text{dec}}$
  - **Bootstrapping** if $d_{hom} \geq d_{dec} + 1$

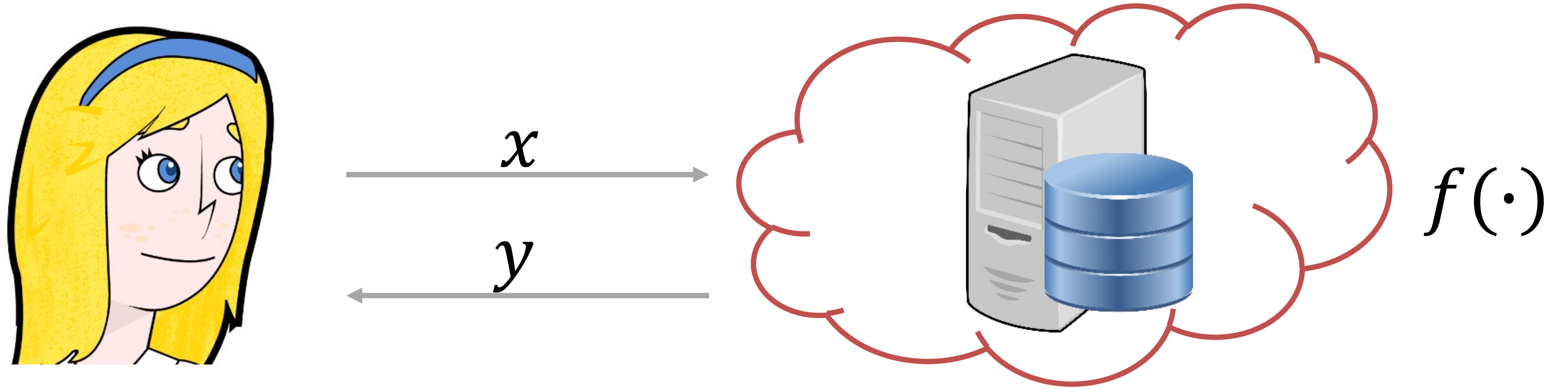


Bootstrapping – Circular Security

- Drawback: Need to generate **many keys**!
- Alternative: **Assume circular security**



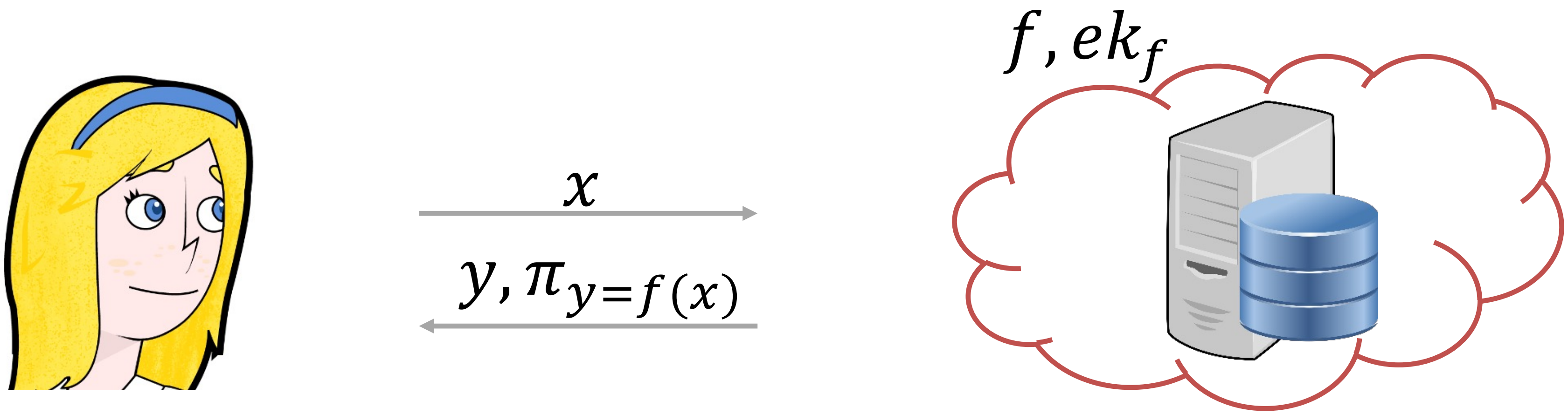
What about Correctness?



$$y =? f(x)$$

- How to verify correctness of the computation?
  - **Without re-computing** the function from scratch
- Important also from the Cloud's perspective
  - Encourage cloud adoption & shed liability

Verifiable Computing



$(ek_f, vk_f) \leftarrow_{\$} \mathbf{G}(f)$

$\mathbf{V}(vk_f, x, y, \pi) \in \{0,1\}$

$y = f(x)$

$\pi_y \leftarrow_{\$} \mathbf{P}(ek_f, x, y)$

Efficiency: Alice's effort **much less** than the effort to compute f

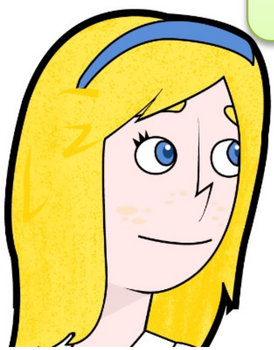
Soundness: No malicious server can cause Alice to accept $y' \neq f(x)$

Verifiable Computing from ABE (1/3)

- Assume an ABE supporting policies $\mathcal{F}$
  - Suffices to take $f \in \mathcal{F}$ to be a **formula**
  - We will need $\mathcal{F}$ to be **closed under complement**

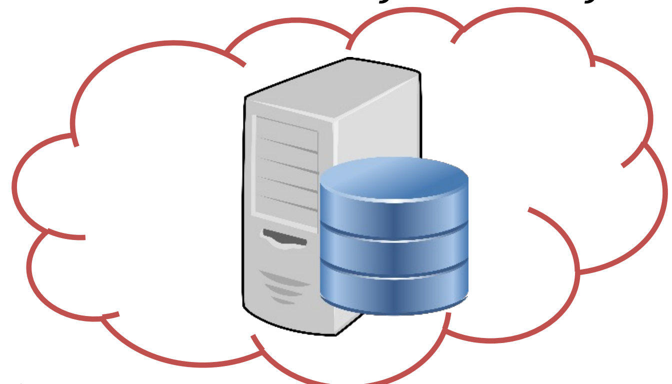
Encryption of random m under attribute x

$f, mpk, ek_f = sk_f$



$c \leftarrow_\$ \mathbf{E}(mpk, x, m)$ →

← d



$f, mpk, vk_f = m$

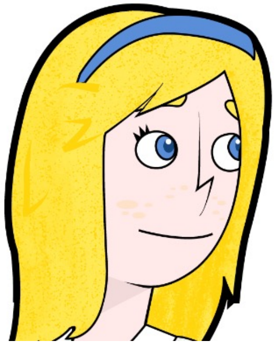
$\mathbf{D}(mpk, sk_f, c) = d$

If $d = m$ conclude that $f(x) = 1$

Verifiable Computing from ABE (2/3)

- The above protocol is a VC scheme for checking that $f(x) = 1$
  - If Alice receives m she is convinced with **no doubt** that $f(x) = 1$ (except with negligible probability)
  - If Alice receives $d \neq m$, we can't conclude that $f(x) = 0$ (as the server could just **refuse to answer**)
  - Hence, the server can cheat only if $f(x) = 1$
- **Idea:** Repeat the protocol **twice**, for $f \in \mathcal{F}$ and for its negation $\bar{f} \in \mathcal{F}$

Verifiable Computing from ABE (3/3)



$mpk, \overline{mpk}$

$vk_f = m, \bar{m}$

$c \leftarrow_\$ \mathbf{E}(mpk, x, m)$

$\bar{c} \leftarrow_\$ \mathbf{E}(\overline{mpk}, x, \bar{m})$

$d, \bar{d}$

If $d = m, y = 1$

If $\bar{d} = \bar{m}, y = 0$

Else, $y =$ error

$ek_f = sk_f, sk_{\bar{f}}$

$mpk, \overline{mpk}$

$\mathbf{D}(mpk, sk_f, c) = d$

$\mathbf{D}(\overline{mpk}, sk_{\bar{f}}, \bar{c}) = \bar{d}$

- For functions with **multi-bit output**, repeat the above for **each function** f_i, where $f_i(x)$ outputs the ith bit of $f(x)$

Additional Properties

- **Public delegatability**
  - Allow arbitrary parties to submit inputs for delegation
  - This is true for any reasonable ABE
- **Public verifiability**
  - Allow arbitrary parties (and not just the delegator) to verify the correctness of the result produced by the worker
  - Can be achieved by publishing $g(m)$ and $g(\overline{m})$, where $g(\cdot)$ is a OWF

ABE from LWE

- It remains to construct an ABE for **expressive enough** policies $\mathcal{F}$
- We sketch a scheme for the class $\mathcal{F}$ of **all Boolean circuits**, based on LWE
  - Let P be the policy circuit with depth d and attribute size k
  - Ciphertext size will be $\text{poly}(k, d)$
  - Key size will be $|sk_P| = |P| + \text{poly}(k, d)$

Main Idea: Key Homomorphism

$$\mathbf{E}(mpk, x, m)$$

P, x

$$\mathbf{E}(pk_P, P(x), m)$$

sk_P

Get m iff $P(x) = 0$

Step 1: Transforming keys

$$mpk = (A, A_1, \dots, A_k)$$

LWE matrices

$$pk_P = A_P = \text{"Compute } P \text{ on } A_1, \dots, A_k\text{"}$$

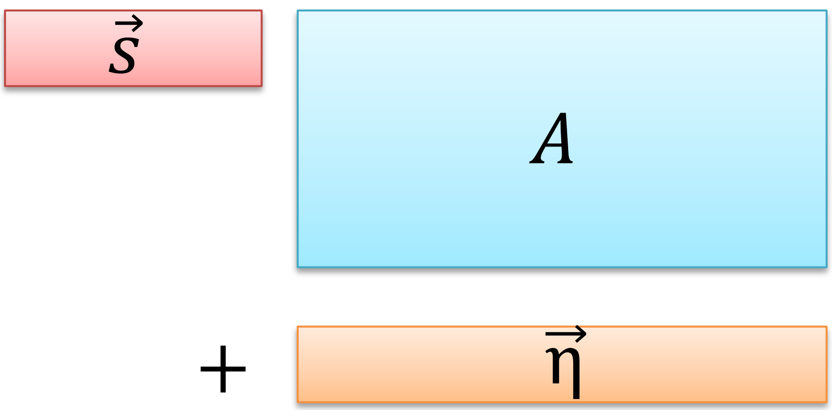
A_u A_v → $+$ → $A_w = A_u + A_v$

G = Gadget matrix

A_u A_v → $\times$ → $A_w = -A_u \times G^{-1}(A_v)$

Step 2: Encryption

$$\mathbf{E}(mpk, x, m) = (\vec{s} \times A + \vec{\eta}, \vec{s} \times (A_1 + x_1 \cdot G) + \vec{\eta}_1, \ldots, \vec{s} \times (A_k + x_k \cdot G) + \vec{\eta}_k, h(\vec{s}) \oplus m)$$



Hard-core bit of randomness s

Step 3: Transforming Ciphertexts (1/2)

$$\mathbf{E}(mpk, x, m) = (\vec{s}\times A + \vec{\eta}, \vec{s}\times(A_1 + x_1 \cdot G) + \vec{\eta}_1, \ldots, \vec{s}\times(A_k + x_k \cdot G) + \vec{\eta}_k)$$

P, x

$$\vec{c}_u = \vec{s}\times(A_u + x_u \cdot G) \qquad \vec{c}_v = \vec{s}\times(A_v + x_v \cdot G)$$

$+$

$$\vec{c}_w = \vec{c}_u + \vec{c}_v = \vec{s}\times((A_u + A_v) + (x_u + x_v) \cdot G)$$

$$\mathbf{E}(pk_P, P(x), m) = (\vec{s}\times A + \vec{\eta}, \vec{s}\times(A_P + P(x) \cdot G) + \vec{\eta}_P$$

Step 3: Transforming Ciphertexts (2/2)

$$\mathbf{E}(mpk, x, m) = (\vec{s}\times A + \vec{\eta}, \vec{s}\times(A_1 + x_1 \cdot G) + \vec{\eta}_1, \ldots, \vec{s}\times(A_k + x_k \cdot G) + \vec{\eta}_k)$$

$$\vec{c}_u = \vec{s}\times(A_u + x_u \cdot G) \qquad \vec{c}_v = \vec{s}\times(A_v + x_v \cdot G)$$

$\times$

$$\vec{c}_w = -\vec{c}_u \times G^{-1}(A_v) + x_u \cdot c_v$$
$$= -\vec{s}\times(A_u \times G^{-1}(A_v) + x_u \cdot A_v) + x_u \cdot \vec{s}\times(A_v + x_v \cdot G)$$
$$= \vec{s}\times((-A_u \times G^{-1}(A_v)) + (x_u \cdot x_v) \cdot G)$$
$$= \vec{s}\times((A_u \times A_v) + (x_u \cdot x_v) \cdot G)$$

Step 4: Decryption

$$\mathbf{E}(mpk, x, m)$$
$$= (\vec{s} \times A + \vec{\eta}, \vec{s} \times (A_1 + x_1 \cdot G) + \vec{\eta}_1, \ldots, \vec{s} \times (A_k + x_k \cdot G) + \vec{\eta}_k)$$

P, x

The secret key sk_P is a **trapdoor** for $A||A_P$ (it allows to compute s and thus m)

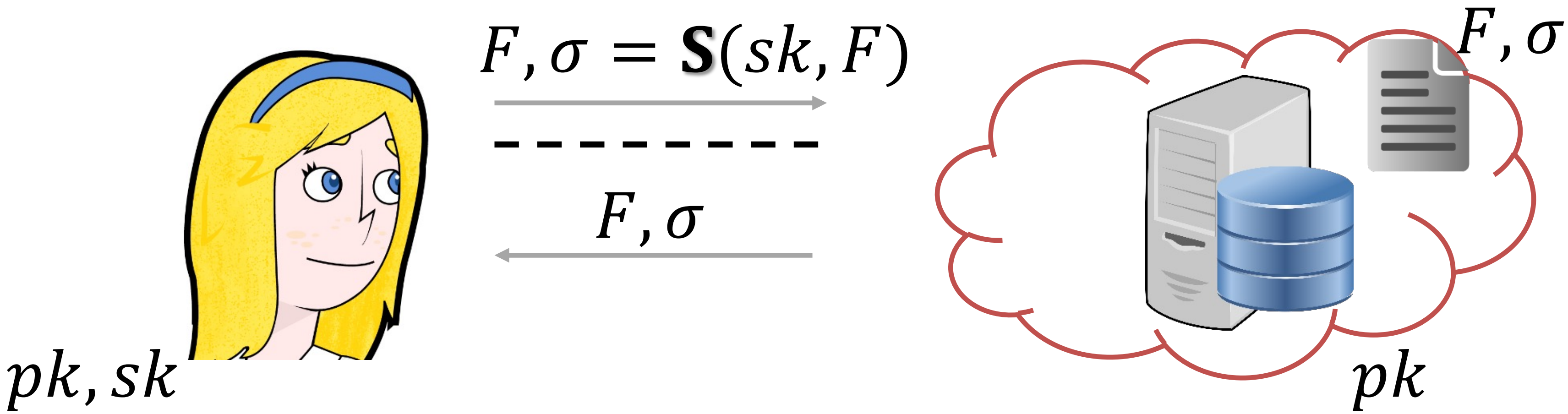
$$\mathbf{E}(pk_P, P(x), m)$$
$$= (\vec{s} \times A||A_P + P(x) \cdot G) + \vec{\eta} + \vec{\eta}_P$$

Cloud Storage

- Lots of data
- Lots of devices
- Wants to access **all data** at **all times** from **all devices**

- Provides **greater accessibility** and reliability
- Cheap price

Naive Protocols

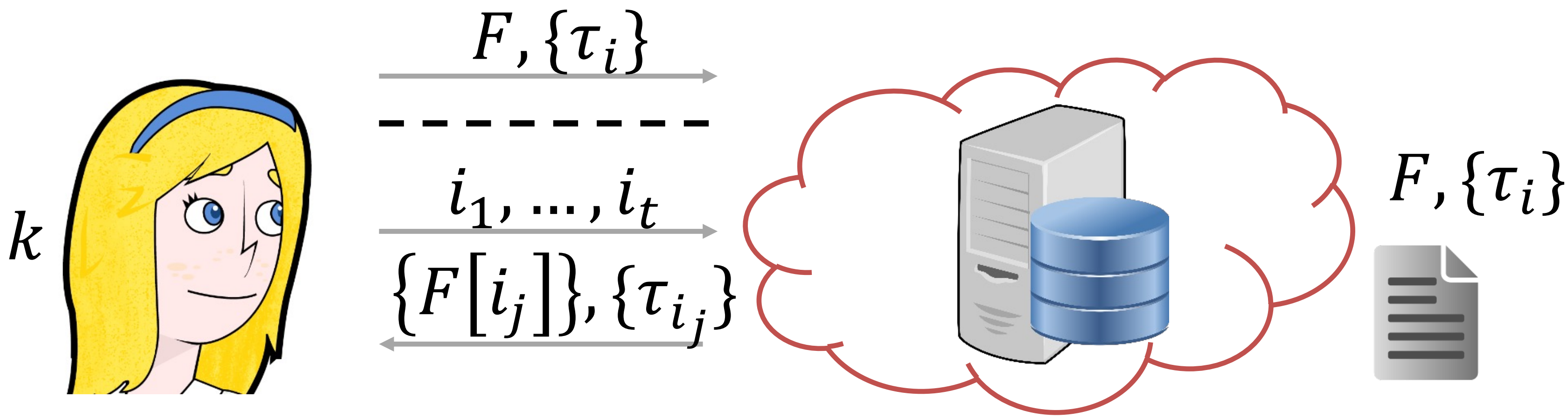


- Run **audit** protocol
- Above protocol is too costly
  - No reason to download all data to run an audit
- What about just checking a **hash** of the file?

Wish List

- **System criteria**
  - Low communication complexity
  - Locality and **small storage** overhead
  - Stateless protocol
- **Crypto criteria**
  - Only an adversary **actually storing** the file can pass an audit
  - Possible to **extract the file** via black-box access
  - Similar to the concept of proof of knowledge

Basic Idea

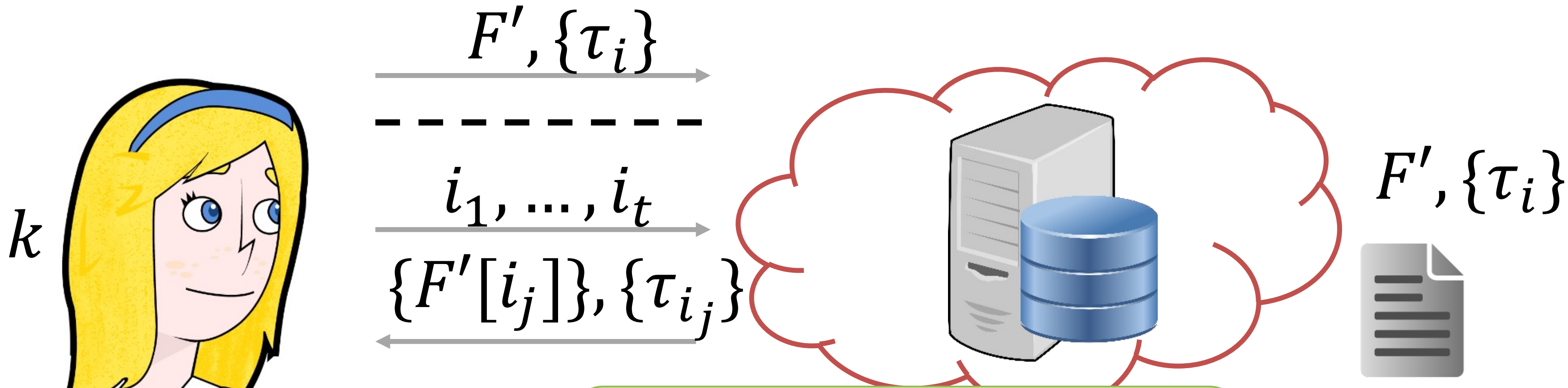


$$F = (F[1], \ldots, F[n])$$

$$\tau_i \leftarrow_{\$} \mathbf{T}(k, F[i])$$

- But server can still **forget** $o(1)$ fraction of blocks and pass audit with **good probability**
  - Pr[detect 1-in-10^6 erasures] < 0.01%
  - Pr[detect 50% erasures]: $1 - (1/2)^t$

Proofs of Retrievability



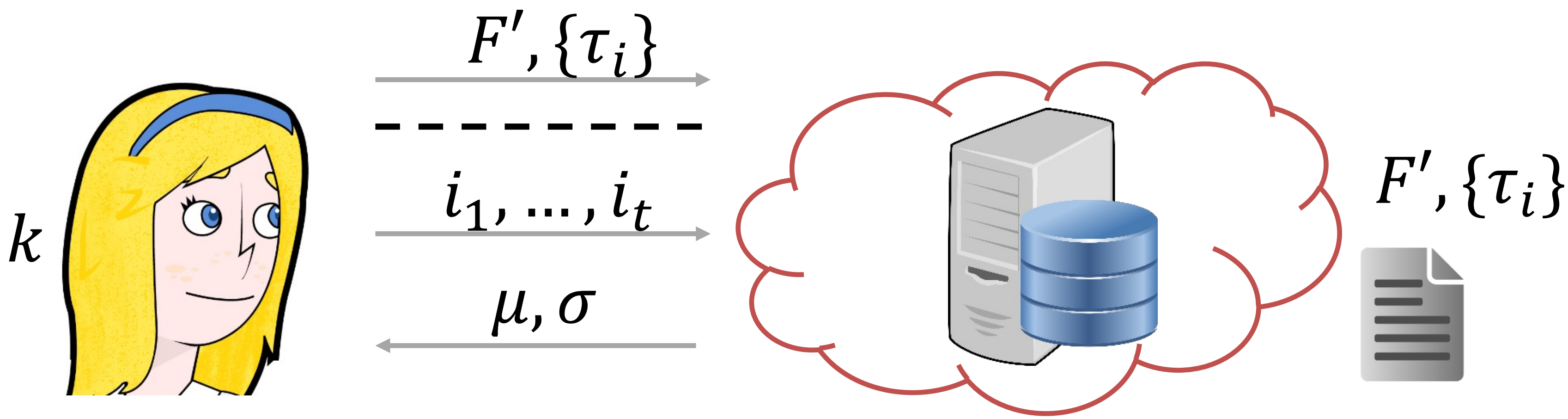
$F' = \mathbf{ECC}(F)$

$\tau_i \leftarrow_{\$} \mathbf{T}(k, i || F'[i])$

Can recover F from any δ fraction (e.g., $\delta = 1/2$)

- If cloud **forgets** $\leq (1 - \delta)$-fraction, can still **reconstruct** F
- If cloud **forgets** $> (1 - \delta)$-fraction, will **pass** an audit w.p. $\leq \delta^t$

Reducing Communication Complexity



$$F' = \mathbf{ECC}(F)$$
$$\tau_i \leftarrow_\$ \mathbf{T}(k, i || F'[i])$$

$$\mu = \sum_j F'[i_j], \sigma = \sum_j \tau_{i_j}$$

- Assume the blocks and the tags are element of some **finite field** $\mathbb{F}$
  - So addition is well defined
- But how can Alice verify?

Homomorphic Authenticators

- Let $\mathbf{PRF}_k: \{0,1\}^* \rightarrow \mathbb{F}; F[i] \in \mathbb{F}; \mathbb{F} = GF(2^{80})$
- **Key:** Single PRF key k and random $\alpha \in \mathbb{F}$
- **Tag:** Compute $\tau_i = \mathbf{PRF}_k(i) + \alpha \cdot F[i]$
- **Aggregate:**

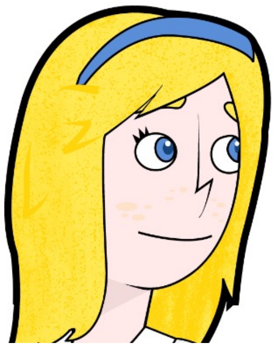
$$\sigma = \sum_i \gamma_i \cdot \tau_i\, ; \mu = \sum_i \gamma_i \cdot F[i]$$

- **Verify:**

$$\sigma = \sum_i \gamma_i \cdot \mathbf{PRF}_k(i) + \alpha \cdot \mu$$

Compact Proofs of Retrievability

k, α



$\{F'[i]\}, \{\tau_i\}$

$Q = \{(i, \gamma_i)\}$ →

$I \subseteq [n]; |I| = t$

$\forall i \in I: \gamma_i \leftarrow_\$ \mathbb{F}$

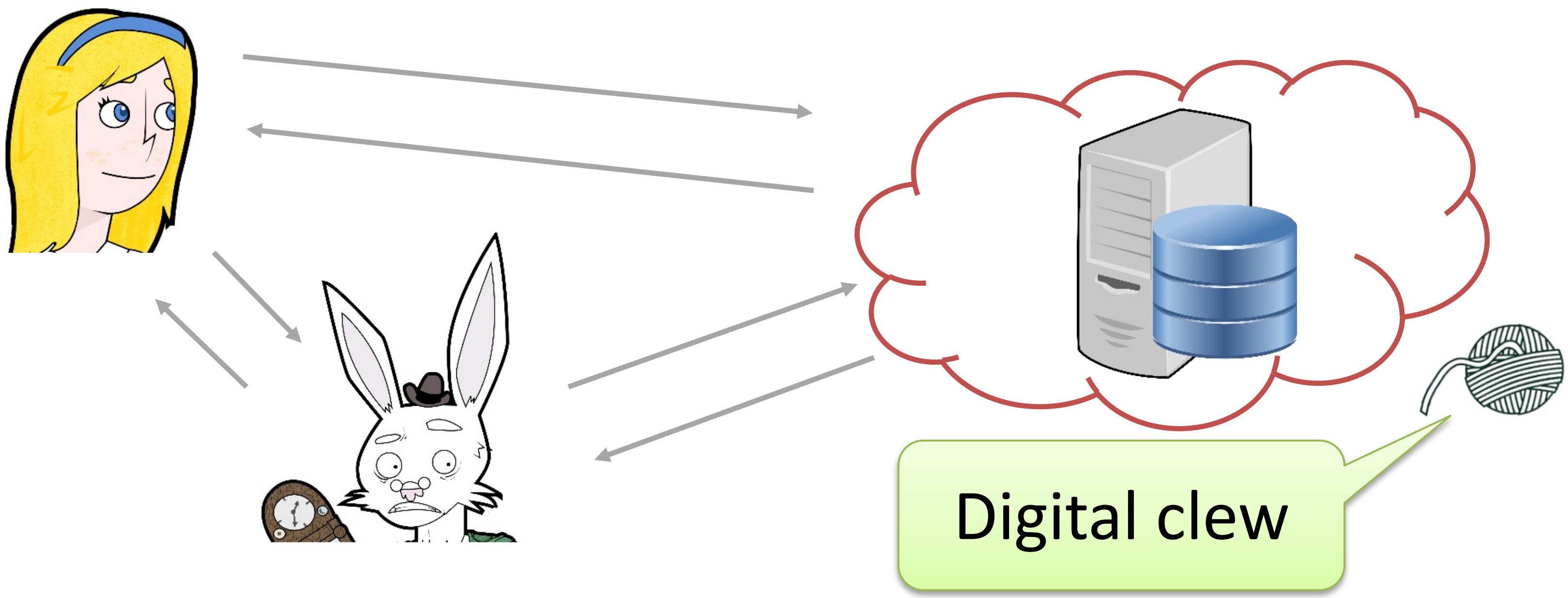
← μ, σ

$$\mu = \sum_{(i,\gamma_i)\in Q} \gamma_i \cdot F'[i]$$

$$\sigma = \sum_{(i,\gamma_i)\in Q} \gamma_i \cdot \tau_i$$

$$\sigma = \sum_{i\in I} \gamma_i \cdot \mathbf{PRF}_k(i) + \alpha \cdot \mu$$

Data Entanglement



- Peer-to-peer approach
- **<u>All-or-nothing integrity:</u>** If cloud forgets a **significant amount** of information, **nobody** will be able to recover its file